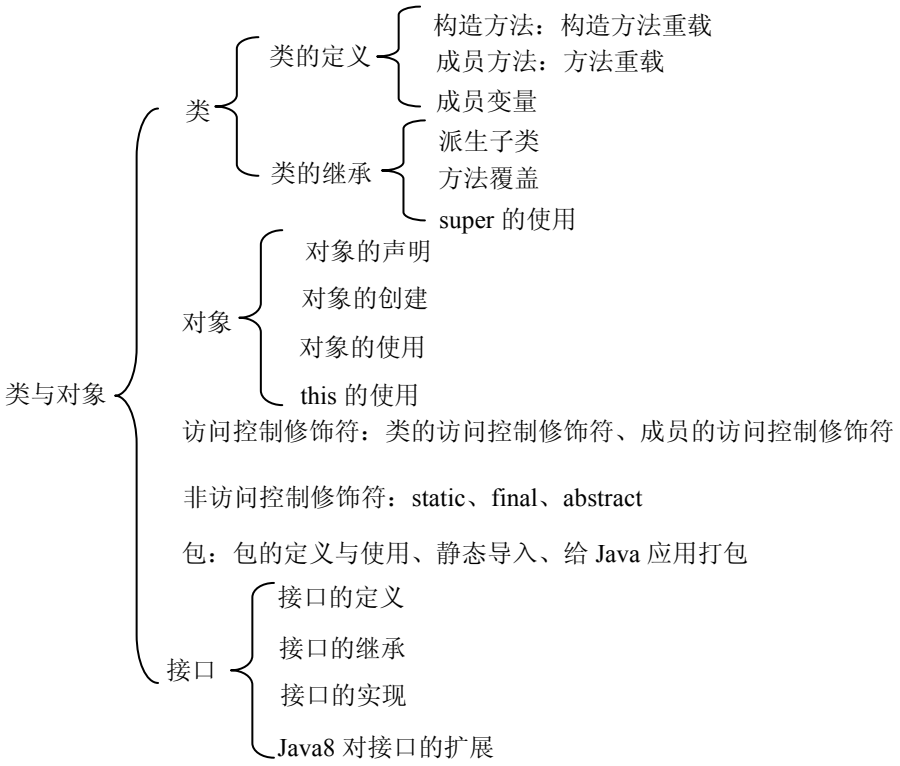
第3章 类与对象

3.1 基本知识提要

3.1.1 知识结构图



3.1.2 重点知识整理

1. 类

类是 Java 语言程序设计的基本单位，Java 程序是由类构成的，编写 Java 程序主要就

是定义各种类。类定义了同一类型对象的结构，定义一个类相当于定义了一个新的“数据类型”，用来规定具体对象的数据域和操作数据的方法。类定义的语法格式为：

```
[<修饰符>] class <类名>[extends<父类名>] [implements<接口名>] {
    [成员变量声明]
    [构造方法定义]
    [成员方法定义]   //类体}
```

1）成员变量

成员变量用来存储对象的属性信息。成员变量可以是 Java 中的任何一种数据类型，包括基本类型和引用类型。成员变量的作用范围是在整个类体内都有效，其有效性与它在类体中定义的先后位置无关，通常先定义成员变量，后定义成员方法。对于成员变量如果不赋初始值，系统会自动赋一个默认值（数值型为 0，boolean 型为 false，引用型为 null）。

2）构造方法

构造方法也称为构造函数，构造方法是类中的一种特殊的方法，通过调用构造方法为实例（对象）分配存储空间同时完成实例（对象）的初始化工作。构造方法的特点如下：

- 构造方法的方法名与类名相同。
- 构造方法没有返回值，也不能写 void。
- 构造方法是在创建一个对象时，使用 new 操作符来调用，完成对象的初始化工作。

3）成员方法

成员方法可以对类中声明的变量进行操作，即给出算法描述对象所具有的行为。成员方法的定义包括两部分：方法头和方法体,其语法格式如下：

```
返回值类型 方法名（参数列表）{
方法体
}
```

方法的返回值类型可以是 Java 中的任何一种数据类型，描述的是在调用方法之后从方法返回的值，当一个方法不需要返回数据时，返回值类型必须为 void。方法的参数可以是任意的 Java 数据类型。

4）方法重载

在面向对象程序设计语言中，有一些方法的含义相同，但带有不同的参数，这些方法使用相同的名字，这就叫方法重载（overloading）。方法重载的方法参数必须不同，或者是参数的个数不同，或者是参数的类型不同。返回值可以相同，也可以不同。方法重载是实现“多态”的一种方法，目的是同一功能有不同的实现，当然构造方法也可以重载。

2. 对象

当定义一个类时，就是定义了一个新的引用数据类型，可以用这个数据类型来声明这种类型的变量，即实例变量或称为对象变量。

1）对象的声明

如同变量的声明一样，对象被创建和使用前必须进行声明，其语法格式如下：

```
类名  对象名;
```

在对象声明的时候，并没有为对象分配存储空间，只有创建对象时，才为对象分配存储空间。这时候不能引用此对象。

2）对象的创建

创建一个对象有两种方法：

（1）先声明再创建。

- 对象声明：

```
类名 对象名;
```

- 对象创建：

```
对象名=new 类名();
```

（2）一步完成。

```
类名 对象名= new 类名();
```

其中，new 是新建对象操作符。通过 new 调用构造方法实例化一个对象，返回该对象的一个引用，即该对象所在的内存地址。

3）对象的使用

对象被创建，即实例化后，可以通过运算符"."访问自己的属性域，也可以调用创建它的类中的成员方法。其语法格式如下：

```
对象.成员变量;
对象.成员方法([参数列表]);
```

4）this 的使用

Java 中的 this 关键字可能有些难理解，this 表示当前对象，也就是调用方法的对象。在 Java 中当前对象是指当前正在调用类中方法的对象。this 有如下作用：

- 使用 this 解决局部变量与成员变量同名的问题。
- 使用 this 引用属性及调用方法。
- 在构造方法中，用 this 调用另一构造方法。

在构造方法中，通过 this 调用另一个构造方法，其语法格式是：

```
this（参数表);
```

this（参数表）语句必须放在构造方法中的第一句。

3. 类的继承

从已有的类派生出新类，这叫做继承。由继承而得到的类称为子类或派生类，被继承的类为父类或超类。父类包括所有直接或间接被继承的类，Java 支持多层继承，一个父类可以有多个间接子类，但继承是单继承的，也就是 Java 中一个类只能有一个直接父类。

如果子类和父类在同一个包中，那么子类继承了其父类中不是 private 的成员，包括成员变量和成员方法，继承的成员变量和成员方法的访问权限保持不变。子类和父类不在同一包中时，父类中的 private 和 default 访问权限的成员不被子类继承，也就是说，子类只继承父类中的 public 和 protected 访问权限的成员作为子类的成员。

1）派生子类

```
[<修饰符>] class <类名>[extends<父类名>]{
      //类体}
```

这样就在两个类之间建立了继承关系，子类从它的父类中继承可访问的数据域和方法，也可以添加新数据域和方法。

2）方法覆盖

在继承关系中，子类会自动继承父类中定义的方法，但有时在子类中需要对继承的方法进行一些修改，对父类的方法进行重写，即子类定义了与父类中同名的方法，实现对父类方法的覆盖。需要注意的是，在子类中重写的方法需要和父类被重写的方法具有相同的方法名、参数列表及返回值类型，并且在子类中重写的方法不能拥有比父类方法更严格的访问权限。

3）super 的使用

子类声明的成员变量与父类的成员变量的名字相同时，在子类中，父类的成员变量将被隐藏。当子类重写父类的方法后，子类对象将无法访问父类被覆盖的方法，为了解决这个问题，在 Java 中提供了 super 关键字，用于从子类中调用父类的构造方法、成员方法和成员变量。

（1）使用 super 关键字引用父类的成员变量和成员方法，其语法格式如下：

- super.成员变量
- super.成员方法（[参数 1，参数 2…]）

（2）使用 super 关键字调用父类的构造方法，其语法格式如下：

- super([参数 1，参数 2…])

注意：通过 super 调用父类的构造方法的代码必须位于子类构造方法的第一行，并且只能出现一次。

4. 访问控制修饰符

1）类的访问控制修饰符

类的访问控制修饰符有两种：public 和默认。

2）类成员的访问控制修饰符

Java 语言为类成员设置了 4 种访问控制修饰符：public、protected、private 和默认，用来修饰成员变量和成员方法。

- public：public 所修饰的类成员可被所有的类访问，public 指定的访问权限范围最大。
- protected：类中由 protected 修饰的成员可以被该类本身、它的子类（包括同一个包中及不同包中的子类）及同一个包中的其他类访问。
- 默认：类中的成员为默认的访问权限时，成员可以被这个类本身和同一包中的类所访问，即包访问权限。
- private：private 所修饰的类成员只能被这个类本身访问。

5. 非访问控制修饰符

1）static

在一个 Java 类中，可以使用 static 关键字来修饰成员变量，该变量被称为静态变量或

类变量。类变量将变量值存储于类的公用内存。类变量访问的语法格式为：

```
类名.变量名。
```

static 既可以在声明属性时使用，也可以在声明方法时使用。有时我们希望在不创建对象的情况下就可以调用某个方法，换句话说，也就是使用该方法不必和具体对象绑定在一起，这时可在定义方法时使用 static 关键字，这样的方法称为静态方法或类方法。类方法调用的语法格式为：

```
类名.方法名
```

2）final

final 关键字在 Java 中表示的意思是最终，可用于修饰类、变量和方法，由 final 修饰的类、变量和方法具有以下特性：

- 使用 final 修饰的类不能被继承，也就是不能有子类。
- 使用 final 修饰的方法不能被子类重写。
- 使用 final 修饰的变量（成员变量和局部变量）是常量，只能赋值一次，不可以修改。

3）abstract

Java 允许定义方法时不写方法体，不包含方法体的方法称为抽象方法，抽象方法必须使用 abstract 关键字修饰。

当一个类中包含了抽象方法时，该类必须使用 abstract 关键字来修饰，使用 abstract 关键字修饰的类称为抽象类。由于抽象类是不完整的（含有抽象方法），所以抽象类不能被实例化，其作用是提供父类，其他类可以继承它的公共部分。抽象类的每个具体子类都必须为父类的抽象方法提供具体实现，否则该子类依然是抽象类。

6. 包

Java 中引入了包（package）机制，包实际上就是一个文件夹，将编写的类、接口分目录存放，避免了名称冲突。

（1）使用 package 关键字定义包，其语法格式为：

```
package pkg1[.pkg2[.pkg3…]];
```

包的声明语句只能位于 Java 源文件的第一行。

（2）使用 import 语句导入了类或类所在的包，就不必写出类的全名了。import 语句的语法格式为：

```
import 包名.子包名.类名;
```

或

```
import 包名.子包名.*;
```

Java 编译器为所有程序自动引入包 java.lang，因此不必用 import 语句引入 java.lang 包含的类。

下面介绍 Java 语言中的常用包。

- java.lang：此包为基本包，包含 Java 语言的核心类，如 String、Math、System 和 Inter 等常用类都在此包中，使用此包中的类时无须使用 import 语句导入包，系统会自动导入这个包中的所有类。

- java.util：此包为工具包，一些常用的类库、日期操作等都在此包中，如果掌握了此包，则对各种设计思路都可以有很好的理解。
- java.net：包含 Java 网络编程相关的类和接口。
- java.sql：数据库操作包，提供了各种数据库操作的类和接口。
- java.io：包含了 Java 输入、输出有关的类和接口。
- java.awt：包含了用于构建和管理应用程序的图形用户界面的类和接口。
- javax.swing：包含了用于建立图形用户界面的类和接口，此包中的组件相对于 java.awt 包而言是轻量级组件。

（3）import 语句不仅可以导入类，还增加了导入静态方法和静态域的功能。如果一个类中的方法全部是静态方法，则在导入时可以直接使用 import static 的方式导入，导入的语法格式为：

```
import static 包.类.*;
```

（4）给 Java 应用打包。在 JDK 中提供了一个 jar 命令，使用这个命令能够将这些类打包成一个文件，这个文件的扩展名为.jar，被称为 jar 文件（Java Archive File），它是一种压缩文件，也被称为 jar 包。

当使用 jar 包时，只需要在 CLASSPATH 环境变量中包含这个 jar 文件的路径（配置 CLASSPATH 环境变量的方法请查阅第 1 章），Java 虚拟机就能自动在内存中解压这个 jar 文件，根据包名所对应的目录结构去寻找所需要的类。jar 命令的主要参数如下：

- C——创建新的文档。
- V——生成详细的输出信息。
- F——指定文档的文件名。

7. 接口

接口（interface）是 Java 中最重要的概念之一，它可以被理解为一种特殊的“类”，是由常量和抽象方法组成的。接口不是类，而是对类的需求的描述，描述类具有什么功能，但并不给出每个功能的具体实现。

1）接口的定义、继承与实现

接口定义的基本语法格式为：

```
<public><abstract> interface 接口名[extends 父类接口列表]{
<public><static><final>变量名=初值; //静态常量
<public><abstract>返回值 方法名（[参数表]）throws[异常列表];     //抽象方法
}
```

注意：

- 关键字 interface 前默认的修饰符为 public abstract，即便没写也自动设为这种权限。
- 接口通过 extends 继承父接口，接口是多继承，因此关键字 extends 后面是接口列表。
- 类通过 implements 关键字实现接口，一个类可以实现一个或多个接口，这实际上就摆脱了 Java 的单继承局限。
- 接口中定义的成员变量，系统自动为这些成员变量增加 public static final 修饰符，必须在定义时指定默认值。
- 一个类需要实现接口中的全部抽象方法，否则类为抽象类。
- 接口中的所有方法都是抽象的和公有的，即是用 public abstract 修饰的，即便没写也

自动地是这个权限。

2）Java8 对接口的扩展

在 Java8 中对接口进行了改进，允许在接口中定义默认方法，默认方法可以提供方法的实现。接口定义的基本语法格式为：

```
<public><abstract> interface 接口名[extends 父类接口列表]{
<public><static><final>变量名=初值；//静态常量
<public><abstract>返回值 方法名（[参数表]）throws[异常列表]；//抽象方法
 default 返回值 方法名（[参数表]）{//默认方法
     方法体
   }
static 返回值 方法名（[参数表]）{//类方法
     方法体
   }
}
```

在 Java8 中，接口里定义的方法只能是抽象方法、类方法或默认方法，因此如果定义的不是默认方法，那么系统将自动为普通方法增加 abstract 修饰符。如果定义的是默认方法，那么必须用 default 修饰符标记，并提供一个默认实现，即接口中的普通方法不能有方法实现（方法体），但类方法、默认方法都必须有方法实现（方法体）。

3.2 典型题解析

例 3.1　下列哪一项不是构造方法的特点？（　　）

A．构造方法名必须与类同名　　B．构造方法不具有任何返回类型

C．任何一个类都含有构造方法　　D．构造方法的访问控制修饰符只能是 public

例题解析：

根据构造方法的特点可知，A、B、C 三个选项都是构造方法的特点。

答案：D

例 3.2　在以下什么情况下，构造方法会被调用？（　　）

A．类定义时　　B．创建对象时

C．调用对象方法时　　D．使用对象的变量时

例题解析：

Java 中通过调用构造方法创建实例（对象），为实例（对象）分配存储空间同时完成实例（对象）的初始化工作。

答案：B

例 3.3　以下程序的运行结果为（　　）。

```
class Test{
    int num;
    public static void main(String args[]){
```

```
        Test x=new Test();
        if(x==null)
            System.out.println("No existing");
        else
            System.out.println(x.num);
    }
}
```

A．0　　B．null　　C．No existing　　D．编译出错

例题解析：

本例中，Test 类中没有定义构造方法，系统提供无参的构造方法，通过调用无参的构造方法创建了对象 x，对象 x 的成员变量没有显性赋值，因此其值为系统提供的默认值 0。

答案：A

例 3.4　以下程序的运行结果为（　　）。

```
class Test{
    int num;
    public static void main(String args[]){
        Test x;
        if(x==null)
            System.out.println("No existing");
        else
            System.out.println(x.num);
    }
}
```

A．0　　B．null　　C．No existing　　D．编译出错

例题解析：

本例中，实例变量 x 仅仅是声明了，并没有通过 new 调用构造方法创建此实例，即在内存中并没有为实例 x 分配存储空间。

答案：D

如果将语句“Test x;”改为“Test x=null;”，则输出：

```
No existing
```

例 3.5　给定如下代码：

```
class UserInfo{
    String userName;
    int userNumber;
    public UserInfo(String userName){
        this.userName=userName;
    }
    public UserInfo(String userName,int userNumber){
        ____________;
        this.userNumber=userNumber;
```

```
    }
}
```

空白处应填写的代码应该是（　　）。

A．UserInfo(userName,userNumber);　　B．this(userName,userNumber);

C．UserInfo(userName);　　D．this(userName);

例题解析：

在本例中 this 的用法是：在构造方法中通过 this 调用另一个构造方法，其语法格式是

```
this（参数列表）;
```

答案：D

例 3.6　下述（　　）方法是方法重载的正确写法。

A．int sumValue(int a,int b){return a+b;}
　　int sumValue(int a,int b){return a;}

B．int sumValue(int a,int b){return a+b;}
　　float sumValue(int a,int b){return (float)(a+b);}

C．int sumValue(int a,int b){return a+b;}
　　float sumValue(float a,float b){return a+b;}

D．int sumValue(int a,int b){return a+b;}
　　int sumValue(int x,int y){return x+y;}

例题解析：

方法重载是在一个类中或在具有继承关系的类中定义多个同名的方法，但方法参数必须不同，或者是参数的个数不同，或者是参数的类型不同。返回值可以相同，也可以不同，即方法重载不是通过返回值来决定是否是方法重载。

答案：C

例 3.7　试分析下列代码的运行结果。

```
class A{
    int num=123;
    void show(){
        System.out.println("class A");
    }
}
class B extends A{
    double num=123.45;
    void show(){
        System.out.println("class B");
        super.show();
        System.out.println(this.num);
        System.out.println(super.num);
    }
}
class C extends B{
```

```
    char num='n';
    void show(){
        System.out.println("class C");
        super.show();
        System.out.println(this.num);
        System.out.println(super.num);
    }
}
public class Test{
    public static void main(String args[]){
        new C().show();
    }
}
```

例题解析：

在继承关系中对成员的访问遵循就近原则。在子类中访问属性和方法时将优先查找自己定义的属性和方法，如果该成员在本类中存在，则使用本类的；否则，按照继承层次的顺序到其祖先类查找。this 关键字指当前对象，即调用方法的对象。super 关键字用于从子类中调用父类的构造方法、成员方法和成员变量。

答案：程序的运行结果为：

```
class C
class B
class A
123.45
123
n
123.45
```

例 3.8 下列代码在编译时会发生错误，下面哪种修改可以更正错误？（　　）

```
package c1;
public class C1 {
     public int x;
     int y;
     private int z;
     public int m1(){return y;  }
     public int m2(){return z;  }
}
package c2;
import c1.C1;
public class C2 {
    void  aMethod(){
         C1 o=new C1();
          System.out.println(o.x);
          System.out.println(o.y);
```

```
        System.out.println(o.m2());
         }
}
```

A．将类 C2 中的语句“System.out.println(o.y);”改为“System.out.println(o.m1());”

B．将类 C1 中的语句“int y;”改为“private int y;”

C．将类 C1 中的语句“int y;”改为“protected int y;”

D．将类 C2 中的语句“System.out.println(o.m2());”改为“System.out.println(o.z);”

例题解析：

本例中，类 C1 和类 C2 分别位于两个包中，类 C1 中的成员变量 y 具有默认的访问控制修饰符，也就是具有包访问权限，对于在类 C2 中创建的类 C1 的对象 o 来说，其属性 y 是不可见的，而在类 C1 中定义的方法 m1()是公有的。

答案：A

例 3.9　试分析下列代码的运行结果。

```
class A{
    int x=1;
    static int y=2;
    void mm(){
        x=2;
    }
    static void nn(){
        y=3;
    }
}
public class Test{
    public static void main(String args[]){
        A a=new A();
        System.out.println(a.x);
        System.out.println(a.y);
        a.mm();
        a.nn();
        System.out.println(a.x);
        System.out.println(a.y);
        A c=new A();
        System.out.println(c.x);
        System.out.println(c.y);
    }
}
```

例题解析：

static 修饰的成员称为类成员，类成员既可通过类名访问也可通过方法名访问。类成员存放在类的公共存储空间，因此 c.y 的值为 3，即 a.y 修改后的 y 的值。注意：类方法中所访问的成员变量只能是类变量。

答案：程序的运行结果为：

```
1,2
2,3
```

```
1,3
```

例 3.10 下面（　　）是正确的接口。

A．interface A{
```
    void show(){  }
}
```
B．abstract interface A {
```
    abstract void show(){  }
}
```
C．abstract interface A{
```
     show();
}
```
D．interface A extends M1,M2{//假定接口 M1,M2 已定义
```
    void show();
}
```
例题解析：

关键字 interface 前默认的修饰符为 public abstract，即便没写也自动地是这个权限。接口通过 extends 继承父接口，接口是多继承，因此关键字 extends 后面是接口列表。接口中的所有方法都是抽象的和公有的，即是用 public abstract 修饰的，即便没写也自动地是这个权限。

答案：D

例 3.11 下面关于方法覆盖的描述不正确的是（　　）。

A．覆盖的方法一定不能是 private 修饰的

B．要求覆盖和被覆盖的方法必须具有相同的访问权限

C．覆盖的方法不能比被覆盖的方法抛出更多的异常

D．要求覆盖和被覆盖的方法有相同的名字、参数列表以及返回值

例题解析：

本题考查的是对方法覆盖的理解。覆盖是指派生类方法覆盖父类的方法，在使用覆盖时需要注意以下几点。

（1）派生类中的覆盖方法必须要和父类中被覆盖方法有相同的返回值类型、函数名和参数。

（2）父类中被覆盖的方法不能为 private 修饰符修饰，否则，其子类只是定义了另外一个方法，并没有实现对父类中方法的覆盖。

（3）子类方法不能缩小父类方法的访问权限。

（4）子类方法不能抛出比父类方法更多的异常。

（5）子类可以覆盖父类的静态方法，但父类的静态方法不能被子类覆盖为非静态的方法；反之亦然。

答案：D

例 3.12 构造方法、成员变量初始化及静态成员变量初始化三者的先后顺序是_____。

例题解析：

当类第一次被加载时，首先静态变量会初始化，接着编译器会把实例变量初始化为默认值，然后执行构造方法。

Java 程序的初始化一般遵循以下原则：

首先，静态变量优于非静态变量初始化，其中，静态变量只初始化一次，而非静态变量可能会初始化多次；

其次，父类优先于子类进行初始化；

最后，按照成员变量的定义顺序对成员变量进行初始化，在构造方法调用之前被初始化。

答案：静态成员变量初始化、成员变量初始化、构造方法。

例 3.13　试分析下列代码的运行结果。

```
1.class B extends A{
2.  int k=10;
3.  public void sum(int x){
4.      super.sum(x);
5.      System.out.println(k);
6.  }
7.}
8.public class A {
9.  int k=5;
10. public void sum(int x){
11.     k=x;
12. }
13  public static void main(String args[]){
14.     new B().sum(1);
15. }
16.}
```

例题解析：

本题考查的是子类中含有覆盖方法的调用情况。子类对象调用方法时,首先调用子类中同名且函数参数形式(参数个数、类型和顺序)一致的方法，然后才调用父类中定义的匹配的方法。本程序运行时，创建了 B 类的对象调用 sum(1)方法，此时程序的执行流程是调用 B 类的 sum()方法，因此输出的 k 值是 B 类对象的成员变量 k 的值。

答案：程序的运行结果为 10。

3.3　测试习题与参考答案

3.3.1　测试习题

一、填空题

1．定义在类中的变量被称为_____，定义在方法中的变量被称为_____。

2．使用对象前需要_____和_____两个步骤。

3．_____是一个特殊的方法，用于初始化一个类的对象。

4．声明为 static 的方法不能访问_____成员。

5．如果需要访问私有数据成员，则通过_____和_____方法实现。

6．Java 中使用关键字_____来创建类的实例对象。

7．被 static 关键字修饰的成员变量被称为_____，它可以被该类所有的实例对象共享。

8．一个类如果实现一个接口，那么它需要实现接口中定义的全部方法，否则该类就必须定义成_____。

9．Java 通过关键字_____可以防止类被继承。

10．_____是 Java 中实现多重继承的唯一途径。

11．接口中定义的数据成员的默认修饰符是_____，接口中定义的成员方法默认修饰符是_____。

12．引用本类中的数据成员和成员方法时，使用_____关键字。

13．在 Java 语言中，要想将一个已编译的类打包成 jar 文件，可以使用的命令是_____。

14．Java 语言提供了两种实现多态的机制，分别是_____和_____。

15.在父类中使用 public 定义的方法，则子类重写该方法的访问控制权限必须是_____。

16．Java 语言中子类继承父类时使用_____关键字，而实现接口时采用_____关键字。

二、选择题

1．下列说法中不正确的是（　　）。

A．局部变量在使用之前无须初始化，因为系统会为该变量提供默认值

B．类成员变量由系统自动进行初始化

C．方法的参数的作用域就是所在的方法

D．语句块中定义的变量，当语句块执行完时，该变量就消亡了

2．下列关于构造方法的描述中错误的是（　　）。

A．Java 语言规定构造方法没有返回值，但不用 void 声明

B．Java 语言规定构造方法名与类名必须相同

C．Java 语言规定构造方法不可以重载

D．Java 语言规定构造方法需要使用 new 关键字调用

3．下列（　　）不是 Java 的 new 操作符的作用。

A．为对象分配内存空间　　B．调用类的构造方法创建对象

C．返回对象的引用　　D．产生一个新的类型

4．访问控制修饰符作用范围由大到小是（　　）。

A．private-protected-default-public　　B．public-protected-default-private

C．private-default-protected-public　　D．public-default-protected-private

5．下面关于关键字 abstract 的描述错误的是（　　）。

A．关键字 abstract 可以修饰类或方法

B．final 类的方法都不能是 abstract，因为 final 类不能有子类
C．abstract 类不能被实例化
D．abstract 类的子类必须实现其超类的所有 abstract 方法

6．在声明类时，声明一个类不能再被继承的关键字是（　　）。
A．unextends　　B．abstract　　C．final　　D．static

7．编译定义了 2 个类和 5 个方法的 Java 源文件后，会产生（　　）字节码文件。
A．2　　B．3　　C．1　　D．5

8．关键字 super 的作用是（　　）。
A．访问父类被隐藏的数据成员　　B．调用父类中被重载的方法
C．调用父类的构造方法　　D．以上选项均是

9．下列说法不正确的是（　　）。
A．Java 语言中允许一个类实现多个接口
B．Java 语言中允许一个类继承多个类
C．Java 语言中允许一个类同时继承一个类并实现多个接口
D．Java 语言中允许一个接口继承多个接口

10．类中的方法被（　　）修饰符修饰时，该方法只能在本类被使用。
A．public　　B．protected　　C．private　　D．default

11．接口的抽象方法都具有（　　）修饰的特性。
A．private，final　　B．public，abstract
C．public，final　　D．protected，abstract

12．下列关于方法重载正确的描述是（　　）。
A．重载方法的返回值类型必须不同
B．重载方法的参数形式不同，即：或者是参数的个数不同，或者是参数的类型不同
C．重载方法的参数名称必须不同
D．重载方法的访问修饰符不同

13．以下关于类的描述正确的是（　　）。
A．只要没有定义不带参数的构造方法，JVM 都会为类生成一个默认的构造方法
B．局部变量的作用范围仅仅在定义它的方法内，或者是定义它的语句块中
C．使用其他类的方法仅仅需要引用方法的名字即可
D．在类中定义的变量称为类的成员变量，在其他类中可以直接使用

14．下列关于继承的描述正确的是（　　）。
A．子类能继承父类的非私有方法和属性
B．子类能继承父类所有的方法和属性
C．子类只能继承父类公有方法和属性
D．子类不能继承父类的 protected 方法和属性

15．下列说法中正确的是（　　）。
A．子类不能定义和父类同名同参数的方法
B．子类只能继承父类的方法，而不能重载

C．重载就是一个类中有多个同名但有不同形参（参数类型或参数个数不同）的方法

D．子类只能覆盖父类的方法，而不能重载父类的方法

16．以下程序段的输出结果为（　　）。

```
interface B{
    int k=1;
}
public class A implements B{
    public static void main(String args[]){
        int i;
        A a=new A();
        i=a.k;
        System.out.println("i="+i);
    }
}
```

A．i=0　　B．程序有编译错误　　C．i=1　　D．i=null

17．以下关于被访问控制符 protected 修饰的成员变量的描述正确的是（　　）。

A．只能被该类自身所访问

B．可以被该类本身和该类的所有子类所访问

C．可以被 3 种类所引用：该类自身、与它在同一个包中的其他类、在其他包中的该类的子类

D．只能被同一包中的类访问

18．下列程序的输出结果是（　　）。

```
class B {
    int k;
    public B(){
    }
    public B(int k){
        this.k=k;
    }
}
public class A extends B{
    public A(){
        k=k+1;
    }
    public static void main(String args[]){
        A a=new A();
        System.out.println(a.k);
    }
}
```

A．输出结果为 1　　B．输出结果为 0

C．输出结果为 2　　D．编译出错

19．下列关于类方法的描述正确的是（　　）。

A．在类方法中可用 this 来调用本类的类方法

B．在类方法中可直接调用本类的类方法

C．在类方法中只能调用本类的类方法

D．在类方法中绝对不能调用实例方法

20．下列关于实例方法的描述正确的是（　　）。

A．实例方法可直接调用超类的类方法

B．实例方法可直接调用超类的实例方法

C．实例方法可直接调用其他类的实例方法

D．实例方法可直接调用本类的类方法

21．已定义接口 X 和 Y，下列语句错误的是（　　）。

A．public interface Z extends X,Y{void aMethod()；}

B．class Z implements X,Y{ }

C．interface Z extends X,Y {void aMethod(){　}；}

D．interface Z extends X{void aMethod()；}

22．已定义接口 B 和 C，以下语句正确的是（　　）。

A．interface A extends B,C

B．interface A implements B,C

C．class A extends B,C

D．class A implements B,implements B

23．下列代码的运行结果是（　　）。

```
public class A{
    public int value(){
        static int i=0;
        i++;
        return i;
    }
    public static void main(String args[]){
        A t=new A();
        System.out.println(t.value());
    }
}
```

A．输出结果为 1　　B．输出结果为 0

C．输出结果为 2　　D．编译出错

24．类 A 是类 B 的子类，类 B 是类 C 的子类，3 个类都定义了方法 method，下列语句（　　）可以在类 A 的方法中调用类 C 的 method 方法。

A．method();　　B．super.method();

C．new C().method();　　D．new A().method();

25．子类调用父类的构造方法时，super（参数列表）语句应该位于（　　）。

A．子类构造方法中的第一句　　B．子类中任意位置

C．子类类体的第一句　　D．子类构造方法中的任意位置

26．定义 Student 类的语句如下：

```
class Student{
    String name,department;
    int score;
    public Student(String name){
        this.name=name;
    }
    public Student(String name,String department){
        this(name);
        this.department=department;
    }
    public Student(String name,String department,int score){
         _______;//调用第二个构造方法
      this.score=score;
      }
}
```

要在第三个构造方法中调用第二个构造方法，空白处的代码应该是（　　）。

A．Student(name,department);　　B．this(String name,String department);

C．this(name,department);　　D．Student(String name,String department);

27．关于 final 关键字的说法错误的是（　　）。

A．修饰的成员变量为常量，它只能被赋值一次，其值不能修改，并且在声明时必须初始化

B．如果修饰类，则该类不能被继承

C．如果修饰方法，则该方法不能被其子类覆盖

D．修饰局部变量时，其值可多次修改

28．下列关于 package 和 import 语句的描述中，错误的是（　　）。

A．一个源文件中 package 语句可以出现一次或多次

B．一个源文件中 import 语句可以出现一次

C．一个源文件中 import 语句必须出现在第一行（不包括注释）

D．一个源文件中 package 语句必须出现在第一行（不包括注释）

29．Java 语言接口的修饰符可以为（　　）。

A．static　　B．protected　　C．final　　D．abstract

30．关于下面的代码，以下结论（　　）是正确的。

```
public class Test {
    public Test(){
        System.out.print("构造方法");
    }
    public void Test(){
```

```
        System.out.print("成员方法");
    }
    public static void main(String args[]){
        Test t=new Test();
        t.Test();
    }
}
```

A．程序无法通过编译

B．程序可以通过编译并正常运行，运行结果为输出：构造方法成员方法

C．程序通过编译但运行出错

D．程序可以通过编译并正常运行，运行结果为输出：构造方法

三、判断题

1．一个类中定义了有参的构造方法，则 Java 系统就不提供默认的构造方法。（　　）

2．方法覆盖要求覆盖的方法和被覆盖的方法必须具有相同的访问权限。（　　）

3．子类可以继承父类的所有成员变量和成员方法。（　　）

4．Java 类中不能定义同名的两个成员方法。（　　）

5．final 修饰的类可以被继承。（　　）

6．每个类都只能有一个构造方法。（　　）

7．static 修饰的类变量为类的所有对象共享。（　　）

8．static 修饰的类方法既可被对象调用，也可通过类名调用。（　　）

9．protected 修饰的成员可以被 3 种类所引用，该类本身、与它在同一个包中的其他类、在其他包中该类的子类。（　　）

10．成员变量无须初始化，可由系统自动进行初始化。（　　）

11．关键字 super 在子类对象中指代其父类对象的引用。（　　）

12．在 Java 中关键字 this 表示当前对象，也就是正在调用类中方法的对象。（　　）

13．类由两部分组成：一部分是变量的定义，一部分是方法的定义。（　　）

14．在定义类时，类的构造方法、成员变量和成员方法的定义顺序在语法上有严格的限制。（　　）

15．类的实例对象的生命周期包括对象的创建、使用、废弃、垃圾回收。（　　）

16．package 语句只能位于 Java 源程序除注释之外的第一行位置。（　　）

17．接口可继承多个接口，创建一个类也可以实现多个接口。（　　）

四、程序阅读题

1．下面程序通过在类中定义方法来实现对私有变量的操作，请将程序补充完整。

```
class A{
int a;
public int b;
private int c;
void setc(int i) {
        ____①____;
}
```

```
int getc() {
        ___②___;
}
}
class Test {
public static void main(String args[]) {
        ___③___ object = new A();
        object.a = 10;
        object.b = 20;
        object.___④___(100);
        System.out.println("a,b,and c: " + object.a + " " +object.b + "
    " + object.___⑤___);
}
}
   }
```

2. 下列哪行代码出现了错误？为什么？

```
1.public class Test {
2.  public static void main(String args[]){
3.      A a=new A();
4.      a.print();
5.  }
6.}
7.class A{
8.  String s;
9.  A(String s){
10.     this.s=s;
11. }
12. public void print(){
13.     System.out.println(s);
14. }
15.}
```

3. 下列哪行代码出现了错误？为什么？

```
1.public class Test {
2.  public static void main(String args[]){
3.      A obj=new A();
4.      System.out.println(obj.i);
5.      System.out.println(obj.s);
6.      obj.aMethod();
7.      obj.bMethod();
8.      System.out.println(A.i);
9.      System.out.println(A.s);
10.     A.aMethod();
11.     A.bMethod();
```

```
12. }
13.}
14.class A{
15. int i;
16. static String s;
17. void aMethod(){
18.         }
19. static void bMethod(){
20.         }
21.}
```

4. 下列哪行代码出现了错误？为什么？

```
1.public class Test {
2.  public static void main(String args[]){
3.      method1();
4.  }
5.  public void method1(){
6.      method2();
7.  }
8.  public static void method2(){
9.      System.out.println("圆的半径是多少："+c.getRadius());
10. }
11. Circle c=new Circle(5);
12.}
13.class Circle{
14. private double radius;
15. public void setRadius(double radius){
16.     this.radius=radius;
17. }
18. public double getRadius(){
19.     return radius;
20. }
21. public Circle(double radius){
22.     this.radius=radius;
23. }
24.}
```

5. 下列代码实现方法覆盖，请将程序补充完整。

```
public class Circle {
    private double radius;
    public Circle(double radius){
        this.radius=radius;
    }
    public double getArea(){
        return 3.14*radius*radius;
```

```
    }
    public double getRadius(){
        return radius;
    }
}
class Cylinder extends Circle{
    private double length;
    public Cylinder(double radius,double length){
        ____①____; //调用父类的构造方法
        this.length=length;
    }
    ____②____ double getArea(){//覆盖父类的方法 getArea
        return length*2*3.14*____③____;//使用父类的成员变量 radius
    }
}
```

6．下列代码中，类 A 和类 B 在不同的包中。如果空白处采用默认的访问权限，类 B 能正常编译吗？如果空白处采用 protected 访问权限，类 B 正常编译吗？

```
package p1;
public class A{
    ____①____int x;
    ____②____void method1(){
}
}
package p2;
import p1.A;
public class B extends A{
    public void method2(){
    System.out.println(x);
    method1();
    }
}
```

7．阅读下面的程序，分析代码是否能通过编译，如果能通过编译，请写出程序运行结果。如果不能通过编译，请说明原因。

```
interface Animal{
    void breathe();
    void run();
    void eat();
}
class Dog implements Animal{
    public void breathe(){
        System.out.println("The dog is breathing!");
    }
    public void eat(){
```

```
        System.out.println("The dog is eating!");
    }
}
class Test{
    public static void main(String args[]){
        Dog dog=new Dog();
        dog.breathe();
        dog.eat();
    }
}
```

五、编程题

1．请按照以下要求设计一个学生类 Student，并进行测试。要求如下：

（1）Student 类中包含两个 private 属性，其中 name 表示姓名，score 表示成绩。

（2）定义两个 set 方法分别用来设置姓名和成绩的属性值，两个 get 方法分别用来获取姓名和成绩的属性值。

（3）定义一个无参的构造方法和一个有两个参数的构造方法（两个参数分别用来为姓名和成绩属性赋值）。

（4）在测试类 Test 中创建两个 Student 对象：一个使用无参的构造方法创建，另一个使用有参的构造方法创建，输出两个对象的属性信息。

2．请按以下要求设计两个类 Circle 和 Cylinder，并进行测试。要求如下：

（1）根据下面的要求设计类 Circle。

Circle 类的成员变量：radius 半径。

Circle 类的方法成员如下所示：

Circle()——无参的构造方法。

Circle(double radius)——有参的构造方法。

double getRadius()——获得圆的半径值。

double getPerimeter()——获得圆的周长。

double getArea()——获得圆的面积。

void disp()——输出圆的信息，包括其半径、周长和面积。

（2）创建类 Cylinder 继承类 Circle，要求如下：

Cylinder 类的成员变量——heitht 表示圆柱的高。

Circle 类的方法成员如下所示：

Cylinder()——无参的构造方法。

Cylinder (double radius,double height)——有参的构造方法。

double getHeight()——获得圆柱的高。

double getArea()——获得圆柱的面积。

double getVol()——获得圆柱的体积。

void disp()——输出圆柱的信息，包括其高、面积和体积。

（3）在测试类 Test 中创建一个 Circle 对象，输出该对象的信息；创建一个 Cylinder 对象，输出该对象的信息。

3．请按以下要求设计一个接口 Shape 及其两个实现类 Circle 和 Square，并进行测试。要求如下：

（1）Shape 接口中有一个抽象方法 getArea()，方法返回一个 double 类型的结果。

（2）Circle 类中定义了一个成员变量 radius，表示半径，实现了接口 Shape 的 getArea()抽象方法，求圆的面积并返回；Square 类中定义了一个成员变量 height，表示边长，实现了接口 Shape 的 getArea()抽象方法，求正方形的面积并返回。

（3）在测试类 Test 中创建 Circle 和 Square 类的对象，计算边长为 5 的正方形面积和半径为 5 的圆的面积。

4．请按以下要求设计一个抽象父类 Person 和它的两个子类 Student 类和 Worker 类，并进行测试。要求如下：父类有学生和工人两个子类，在父类中定义其共同属性，子类学生和工人都可以说话，但学生和工人说话的内容是不一样的，具体内容由学生或工人决定，请利用抽象类来完成这种场景。

3.3.2 参考答案

一、填空题

1．成员变量　局部变量　2．声明对象　创建对象　3．构造方法　4．非静态

5．setter　getter　6．new　7．类变量　8．抽象类　9．final　10．接口

11．final public static　public abstract　12．this　13．jar-cvf　14．重载　覆盖

15．public　16．extends　implements

二、选择题

1．A　2．C　3．D　4．B　5．D　6．D　7．A　8．D　9．B　10．A

11．B　12．B　13．B　14．A　15．C　16．C　17．C　18．A　19．B　20．D

21．C　22．A　23．D　24．C　25．A　26．C　27．D　28．D　29．D　30．B

三、判断题

1．√　2．√　3．×　4．×　5．×　6．×　7．√　8．√　9．√　10．√

11．√　12．√　13．√　14．×　15．√　16．√　17．√

四、程序阅读题

1．①c=i　②reurn c　③ A　④setc　⑤getc()

2．第 3 行代码出现了错误，因为类中声明了一个有参的构造方法，则系统不会提供默认的构造方法。

3．第 8 行和第 9 行代码出现了错误，因为它们不是类成员，不能用类名来引用。

4．第 4 行和第 10 行代码出现了错误。第 4 行的代码错误在于 method1 方法不是静态方法，不能在静态方法中调用实例方法；第 10 行的代码错误在于可以在静态方法中调用实例方法，但此时必须在静态方法中创建实例对象，通过实例对象调用实例方法。

5．①super(radius)　②public　③ getRadius()

6．类 A 和类 B 在不同的包中。如果空白处采用默认的访问权限，则类 B 能不正常编译；如果空白处采用 protected 访问权限，则类 B 能正常编译。

7．不能通过编译。因为当创建类实现接口时必须实现接口中的全部抽象方法，否则

该类是抽象类，本程序中的 Dog 类不是抽象类，但没有实现接口中的全部抽象方法，因此不能通过编译。

五、编程题

1．参考源代码如下：

```
class Student{
    private String name;
    private double score;
    public void setName(String name){
        this.name=name;
    }
    public void setScore(double score){
        this.score=score;
    }
    public String getName(){
        return name;
    }
    public double getScore(){
        return score;
    }
    Student(){}
    Student(String name,double score){
        this.name=name;
        this.score=score;
    }
}
public class Test{
    public static void main(String args[]){
        Student z=new Student();
        z.setName("李丽");
        z.setScore(95);
        System.out.println("学生姓名是："+z.getName()+",学生的成绩是：
        "+z.getScore());
        Student p=new Student("李军",77);
        System.out.println("学生姓名是："+p.getName()+",学生的成绩是：
        "+p.getScore());
        }
}
```

2．参考源代码如下：

```
class Circle{
    private double radius;
    Circle(){
        radius=0;
    }
```

```
    Circle(double radius){
        this.radius=radius;
    }
    double getRadius(){
        return radius;
    }
    double getPerimeter(){
        return 2*3.14*radius;
    }
    double getArea(){
        return 3.14*radius*radius;
    }
    void disp(){
        System.out.println("圆的半径、周长、面积分别是："+getRadius()+"、
        "+getPerimeter()+"、"+getArea());
    }
}
class Cylinder extends Circle{
    private double height;
    Cylinder(){
        super();
        height=0;
    }
    Cylinder(double radius,double height){
        super(radius);
        this.height=height;
    }
    double getHeight(){
        return height;
    }
    double getArea(){
        return getPerimeter()*height;
    }
    double getVol(){
        return super.getArea()*height;
    }
    void disp(){
        System.out.println("圆柱的高、半径、表面积和体积分别是："+getHeight()+"、
        "+getRadius()+"、"+getArea()+"、"+getVol());
    }
}
public class Test{
    public static void main(String args[]){
        Circle c=new Circle(5);
        c.disp();
```

```
        Cylinder cy=new Cylinder(5,10);
        cy.disp();
    }
}
```

3．参考源代码如下：

```
interface Shape{
    double getArea();
}
class Circle implements Shape{
    private double radius;
    Circle(double radius){
        this.radius=radius;
    }
    public double getArea(){
        return 3.14*radius*radius;
    }
}
class Square implements Shape{
    private double height;
    Square(double height){
        this.height=height;
    }
    public double getArea(){
        return height*height;
    }
}
public class Test{
    public static void main(String args[]){
        Circle c=new Circle(5);
        System.out.println("圆的面积是："+c.getArea());
        Square s=new Square(5);
        System.out.println("正方形的面积是："+s.getArea());
    }
}
```

4．参考源代码如下：

```
abstract class Person{
    private String name;
    private int age;
    public Person(String name,int age){
        this.name=name;
        this.age=age;
    }
    public String getName(){
        return name;
```

```
    }
    public int getAge(){
        return age;
    }
    public void say(){
        System.out.println(this.getContent());
    }
    public abstract String getContent();
}
class Student extends Person{
    private float score;
    public Student(String name,int age,float score){
        super(name,age);
        this.score=score;
    }
    public String getContent(){
        return "学生信息："+super.getName()+"这学期的成绩是："+this.score;
    }
}
class Worker extends Person{
    private double salary;
    public Worker(String name,int age,double salary){
        super(name,age);
        this.salary=salary;
    }
    public String getContent(){
        return "工人信息："+super.getName()+"现在的工资是："+this.salary;
    }
}
public class Test{
    public static void main(String args[]){
        Person p=new Student("Lili",18,560f);
        Person q=new Worker("张三",40,5600f);
        p.say();
        q.say();
    }
}
```

3.4 实训 3　面向对象编程练习

3.4.1　类

实训目的

1．通过编程和上机实验理解 Java 语言是如何体现面向对象编程的基本思想，以及如何创建类和对象，掌握成员变量和成员方法的特性。

2．理解构造方法的作用，掌握构造方法的设计。

3．理解方法重载的意义，掌握构造方法重载和方法重载的设计。

实训内容

1．编写 Book 类，在该类中定义了 3 个 private 属性，即 title 表示书名、author 表示作者、price 表示价格；3 个 set 方法分别用来设置书名、作者和价格的值，3 个 get 方法分别用来获取书名、作者和价格。编写测试类 Test，用来测试 Book 类，创建 Book 对象并输出其属性。

2．对下面的程序写出输出结果，并能说明方法重载的意义及语法要求。

```
class SumClass{
    int sum(int a,int b){
        return a+b;
    }
    double sum(double a,double b){
        return a+b;
    }
    int sum(int a,int b,int c){
        return a+b+c;
    }
    double sum(double a,double b,double c){
        return a+b+c;
    }
}
class Test{
    public static void main(String args[]){
        System.out.println(new SumClass().sum(98,99));
        System.out.println(new SumClass().sum(77.5,89.2));
        System.out.println(new SumClass().sum(98,99,88));
        System.out.println(new SumClass().sum(66.5,89.9,67.8));
    }
}
```

参考源代码

1. 参考源代码如下：

```
class Book{
    private String title;
    private String author;
    private double price;
    public void setTitle(String title){
        this.title=title;
    }
    public void setAuthor(String author){
        this.author=author;
    }
    public void setPrice(double price){
```

```
        this.price=price;
    }
    public String getTitle(){
        return title;
    }
    public String getAuthor(){
        return author;
    }
    public double getPrice(){
        return price;
    }
}
public class Test{
    public static void main(String args[]){
        Book book=new Book();
        book.setAuthor("朱自清");
        book.setTitle("荷塘月色");
        book.setPrice(28.5);
        System.out.println("书名是："+book.getTitle()+",书的作者是：
        "+book.getAuthor()+",价格是："+book.getPrice());
    }
}
```

2．答案略，参考 3.1.2 节的介绍。

3.4.2 对象的创建与使用

实训目的

1．理解声明对象与创建对象的区别。

2．学会使用构造方法创建对象，掌握对象的使用。

3．掌握 this 关键字的使用。

实训内容

1．定义一个 Person 类，可以在应用程序中使用该类。Person 类的成员属性（变量）如下：

姓名：name，字符串类型：String；

性别：sex，字符型：char；

年龄：age，整型：int。

3 个构造函数：

```
public Person(String name)          //设置姓名
public Person(String name,char sex)
public Person(String name,char sex,int age)
```

一个成员方法：

```
public String toString()   //获得姓名、性别和年龄
```

利用定义的 Person 类，请实例化对象，输出下面结果：

姓名：张三　　性别：男　年龄：21

把下面程序补充完整，调试通过并写出结果。

```
public class Person {
private String name;
private char sex;
    private int age;
    public Person(String name){
        this.name =name;
    }
    public Person(String name,char sex){
        ___①___;//调用本类的构造函数 Person(String name)
        this.sex = sex;
    }
    public Person(String name,char sex,int age){
        ___②___;//调用本类的构造函数 Person(String name,char sex)
        this.age =age;
    }
    public String toString(){
        return ___③___;
    }
    public static void main(String[] args){
        Person p= ___④___ Person("lili",'女',21);
        System.out.println(p.___⑤___);
    }
}
```

2．设计一个名为 Account 的类模拟账户，它包括：

int 型数据域 id 表示账号（默认值为 0）；

double 型数据域 balance 表示账户余额（默认值为 0）；

double 型数据域 interestRate 表示存储年利率（默认值为 0）；

Date 型数据域 dateCreated 存储账户开户的日期；

用无参构造方法创建一个默认账户；

id，balance，dateCreated 和 interestRate 的 getter 和 setter 方法；

方法 withDraw()用于从账户取钱操作；

方法 deposit()用于向账户存钱操作。

编写一个测试程序，创建一个账号为 102231、余额为 20 000、年利率为 3.31%的 Account 对象。使用 withDraw()方法提款 2000 元，使用 deposit 方法存款 3000 元，输出该账户的账户名、开户日期、利率及余额。

参考源代码

1．①this(name)　②this(name,sex)

③"姓名："+this.name+"，"+"性别："+this.sex+"，"+"年龄："+this.age
④new ⑤toString()

2．参考源代码如下：

```
import java.util.Date;
class Account{
    int id=0;
    double balance=0;
    double interestRate=0;
    Date dateCreated;
    void setId(int id){
        this.id=id;
    }
    void setBalance(double balance){
        this.balance=balance;
    }
    void setInterestRate(double interestRate){
        this.interestRate=interestRate;
    }
    void setDateCreated(Date dateCreated){
        this.dateCreated=dateCreated;
    }
    int getId(){
        return id;
    }
    double getBalance(){
        return balance+balance*getInterestRate();
    }
    double getInterestRate(){
        return interestRate;
    }
    Date getDateCreated(){
        return dateCreated;
    }
    void withDraw(double num){
        if(balance>=num)balance=balance-num;
        else System.out.println("余额不足！");
    }
    void deposit(double num){
        balance=balance+num;
    }
}
class Test{
    public static void main(String args[]){
        Account ac=new Account();
```

```
        ac.setId(102231);
        ac.setBalance(20000);
        ac.setInterestRate(0.0331);
        ac.setDateCreated(new Date());
        ac.withDraw(2000);
        ac.deposit(3000);
        System.out.println("账户名是："+ac.getId()+",开户日期为：
        "+ac.getDateCreated()+",利率是："+ac.getInterestRate()+"，余额为：
        "+ac.getBalance());
    }
}
```

3.4.3　类的继承

实训目的

1．理解类有继承性。

2．利用继承性由父类创建子类，掌握方法重写的使用。

3．理解成员变量、成员方法的继承与隐藏，掌握 super 关键字的使用。

实训内容

定义普通人 Person 类、教师 Teacher 类、学生 Student 类，其中 Teacher 类和 Student 类是 Person 类的子类，Person 类中包含姓名、年龄等属性，以及显示信息等方法。Student 类除了具有 Person 类的属性外，还有自己的属性如课程成绩、学校等，同时有自己的显示信息的方法。Teacher 类除了具有 Person 类的属性外，还有自己的属性如讲授的课程等，同时有自己的显示信息的方法。编写一个测试程序，创建学生和老师对象并显示他们的信息。

参考源代码

参考源代码如下：

```
class Person{
    String name;
    int age;
    void toPrintInfo(){
        System.out.println("这个人的名字是："+name+",年龄是："+age);
    }
}
class Student extends Person{
    String schoolname;
    double score;
    void toPrintInfo(){
        super.toPrintInfo();
        System.out.println("学生的学校是："+schoolname+"成绩是："+score);
    }
}
```

```
class Teacher extends Person{
    String course;
    void toPrintInfo(){
        super.toPrintInfo();
        System.out.println("教师讲授的课程是："+course);
    }
}
public class Test {
    public static void main(String[] args) {
        Student z=new Student();
        z.name="lili";
        z.age=18;
        z.schoolname="三中";
        z.score=465;
        z.toPrintInfo();
        Teacher t=new Teacher();
        t.name="马华";
        t.age=38;
        t.course="数学";
        t.toPrintInfo();
    }
}
```

3.4.4 访问控制修饰符

实训目的

掌握包结构下的类及成员的访问控制权限的设计策略。

实训内容

定义一个 public 类 X 和一个默认访问权限的类 Y，每个类中包含 4 个具有不同访问控制权限的成员变量和 4 个不同访问控制权限的成员方法。分别验证类的两种访问控制权限和类成员的 4 种访问控制权限：

（1）在类 X 中定义主方法 main()，对 X 和 Y 类加以应用，观察程序的编译结果。

（2）在另一个包中创建测试类 Test，其中含有主方法，对 X 和 Y 类加以应用，观察程序编译结果。

参考源代码

（1）参考源代码如下：

```
public class X {
    public String s1="public";
    String s2="default";
    protected String s3="protected";
    private String s4="private";
    public void method1(){
```

```
        System.out.println("public method");
    }
    void method2(){
        System.out.println("default method");
    }
    protected void method3(){
        System.out.println("protected method");
    }
    private void method4(){
        System.out.println("private method");
    }
    public static void main(String[] args) {
        X x=new X();
        System.out.println("变量的公有权限"+x.s1+"变量的默认权限"+x.s2+"变量
        的保护权限"+x.s3+"变量的私有权限"+x.s4);
        x.method1();x.method2();x.method3();x.method4();
        Y y=new Y();
        System.out.println("变量的公有权限"+y.str1+"变量的默认权限"+y.str2+"
        变量的保护权限"+y.str3+"变量的私有权限"+y.str4);
        y.m1();y.m2();y.m3();y.m4();
    }
}
class Y{
    public String str1="public";
    String str2="default";
    protected String str3="protected";
    private String str4="private";
    public void m1(){
        System.out.println("public method");
    }
    void m2(){
        System.out.println("default method");
    }
    protected void m3(){
        System.out.println("protected method");
    }
    private void m4(){
        System.out.println("private method");
    }
}
```

X 类和 Y 类在同一个包中，由于主方法是定义在 X 类中，因此在主方法中对于 X 的对象的所有属性和方法均可访问，而对于具有默认访问权限的类 Y 的对象的私有属性和方法不可访问。

（2）参考源代码如下：

```
package p1;
public class X {
    public String s1="public";
    String s2="default";
    protected String s3="protected";
    private String s4="private";
    public void method1(){
        System.out.println("public method");
    }
    void method2(){
        System.out.println("default method");
    }
    protected void method3(){
        System.out.println("protected method");
    }
    private void method4(){
        System.out.println("private method");
    }
}
class Y{
    public String str1="public";
    String str2="default";
    protected String str3="protected";
    private String str4="private";
    public void m1(){
        System.out.println("public method");
    }
    void m2(){
        System.out.println("default method");
    }
    protected void m3(){
        System.out.println("protected method");
    }
    private void m4(){
        System.out.println("private method");
    }
}
package p2;
import p1.*;
public class Test {
    public static void main(String[] args) {
        X x=new X();
        System.out.println("变量的公有权限"+x.s1+"变量的默认权限"+x.s2+"变量
```

```
            的保护权限"+x.s3+"变量的私有权限"+x.s4);
            x.method1();x.method2();x.method3();x.method4();
            Y y=new Y();
            System.out.println("变量的公有权限"+y.str1+"变量的默认权限"+y.str2+"
            变量的保护权限"+y.str3+"变量的私有权限"+y.str4);
            y.m1();y.m2();y.m3();y.m4();
        }
    }
```

X 类和 Y 类在同一个包 p1 中，具有主方法的测试类 Test 在包 p2 中，对于具有公有访问权限的类 X 的对象，其公有属性和方法均可访问，而其余的属性和方法不可访问。而对于具有默认访问权限 Y 的对象的所有属性和方法均不可访问，因为具有默认访问权限的类的作用范围为同一包中。

3.4.5　非访问控制修饰符

实训目的

1．理解 static 的使用。

2．理解抽象方法和抽象类的使用。

实训内容

1．阅读以下程序，通过两个类说明静态变量与实例变量、静态方法与实例方法的区别。

```
class StaticDemo {
static int x;
int y;
public static int getX() {
return x;
}
public static void setX(int newX) {
x = newX;
}
public int getY() {
return y;
}
public void setY(int newY) {
y = newY;
}
}
public class Test {
public static void main(String[] args) {
System.out.println("静态变量 x="+StaticDemo.getX());
System.out.println("实例变量 y="+StaticDemo.getY()); //编译时将出错
StaticDemo a= new StaticDemo();
```

```
StaticDemo b= new StaticDemo();
a.setX(1);
a.setY(2);
b.setX(3);
b.setY(4);
System.out.println("静态变量 a.x="+a.getX());
System.out.println("实例变量 a.y="+a.getY());
System.out.println("静态变量 b.x="+b.getX());
System.out.println("实例变量 b.y="+b.getY());
}
}
```

2．定义一个抽象父类 Shape 表示形状，它包含一个抽象方法 getArea()，从 Shape 类派生出 Rectangle 和 Circle 类表示矩形和圆，这两个类都用 getArea()方法计算对象的面积。编写应用程序使用 Rectangle 和 Circle 类。

参考源代码

1．此题注意类变量和类方法的访问方式。非 static 声明的方法可以调用类方法或类变量，但类方法不能调用非静态的变量或方法。

2．参考源代码如下：

```
abstract class Shape{
    abstract double getArea();
}
class Rectangle extends Shape{
    double height;
    double length;
    Rectangle(double height,double length){
        this.height=height;
        this.length=length;
    }
    double getArea(){
        return height*length;
    }
}
class Circle extends Shape{
    double radius;
    Circle(double radius){
        this.radius=radius;
    }
    double getArea(){
        return 3.14*radius*radius;
    }
}
public class Test{
    public static void main(String args[]){
```

```
        Rectangle r=new Rectangle(5,10);
        System.out.println("矩形的面积是："+r.getArea());
        Circle c=new Circle(3);
        System.out.println("圆的面积是："+c.getArea());
    }
}
```

3.4.6　包

实训目的

1．理解 Java 中包的组织结构和定义方法。

2．使用 jar 命令，创建可执行的 jar 包。

实训内容

定义普通人、教师、学生这些类，提供适当的成员变量、方法用于描述其内部数据和行为特征，并提供主类使之运行。要求有良好的封装性，将不同类放在不同的包下面，创建可执行的 jar 包。

参考源代码

参考源代码如下（假设代码在同一项目中）：

```
package p1;
public class Person {
    String id;
    String name;
    int age;
    public Person(){}
    public Person(String id,String name,int age){
        this.id=id;
        this.name=name;
        this.age=age;
    }
    protected void printInfo(){
        System.out.println("这个人的身份证号是："+id+",名字："+name+",年龄：
        "+age);
    }
}
package p2;
import p1.Person;
public class Teacher extends Person{
    double salary;
    String course;
    public Teacher(double salary,String course){
        this.salary=salary;
        this.course=course;
    }
```

```
    public Teacher(String id,String name,int age,double salary,String
    course){
        super(id,name,age);
        this.salary=salary;
        this.course=course;
    }
    public void printInfo(){
        super.printInfo();
        System.out.println("薪水："+salary+",课程："+course);
    }
}
package p3;
import p1.Person;
public class Student extends Person{
    String schoolname;
    double score;
    public Student(String schoolname,double score){
        this.schoolname=schoolname;
        this.score=score;
    }
    public Student(String id,String name,int age,String schoolname,double
    score){
        super(id,name,age);
        this.schoolname=schoolname;
        this.score=score;
    }
    public void printInfo(){
        super.printInfo();
        System.out.println("学校是："+schoolname+",成绩："+score);
    }
}
package p;
import p2.Teacher;
import p3.Student;
public class Test {
    public static void main(String[] args) {
        Teacher t=new Teacher("2301111","马华",38,3000,"数学");
        t.printInfo();
        Student s=new Student("2311010","lili",16,"三中",620);
        s.printInfo();
    }
}
```

将 Person 类打包成 jar 文件，以及使用 jar 包的操作：

打开命令提示符，进入 d:\workspace\K\src 目录，输入命令：

```
java-cvf Person.jar p1
```

上面的命令用于把 p1 目录下的全部内容生成一个 Person.jar 文件。注意，如果当前目录下已经存在 Person.jar 文件，那么该文件将被覆盖。在使用 jar 包时，只需要在 classpath 环境变量中包含这个 jar 文件的路径，Java 虚拟机就能自动在内存中解压这个 jar 文件，根据包名对应的目录结构去寻找所需要的类。

3.4.7　接口

实训目的

1．了解接口的继承性。

2．掌握接口的实现与运用方法。

3．巩固 Java 的多态性与继承性的应用。

实训内容

设计接口 Vehicle 表示运输工具，有移动（move）、加速（speedUp）、减速（slowDown）、停止（stop）等方法。设计汽车 Automobile、货车 Van、轿车 Car 等类的实现接口 Vehicle，其中 Automobile 是抽象父类，有名称（name）、最大载重量（loadCapacity）、最高速度（maxSpeed）等属性，Van 和 Car 是 Automobile 的子类，同时要求设计主类使程序运行。该实验内容要求涵盖接口、抽象类、继承等核心知识点。

参考源代码

参考源代码如下：

```
interface Vehicle{
    void move();
    void speedUp();
    void slowDown();
    void stop();
}
abstract class Automobile implements Vehicle{
    String name;
    int loadCapacity;
    int maxSpeed;
    public void move(){
        System.out.println("是汽车在公路上行驶！");
    }
}
class Van extends Automobile{
    public Van(String name,int loadCapacity,int maxSpeed){
        this.name=name;
        this.loadCapacity=loadCapacity;
        this.maxSpeed=maxSpeed;
    }
    public void speedUp(){
```

```
        System.out.println("是 Van 在加速！");
    }
    public void slowDown(){
        System.out.println("是 Van 在减速！");
    }
    public void stop(){
        System.out.println("是 Van 在制动！");
    }
}
class Car extends Automobile{
    public Car(String name,int loadCapacity,int maxSpeed){
        this.name=name;
        this.loadCapacity=loadCapacity;
        this.maxSpeed=maxSpeed;
    }
    public void speedUp(){
        System.out.println("是 Car 在加速！");
    }
    public void slowDown(){
        System.out.println("是 Car 在减速！");
    }
    public void stop(){
        System.out.println("是 Car 在制动！");
    }
}
public class Test {
    public static void main(String[] args) {
        Van v=new Van("货车",6,100);
        v.move();v.speedUp();v.slowDown();v.stop();
        Car c=new Car("轿车",4,120);
        c.move();c.speedUp();c.slowDown();c.stop();
    }
}
```

3.5 拓展训练

1. 一个子类继承父类，运行程序可以观察父类与子类的初始化顺序。

```
class Art{
    Art(){
        System.out.println("Art constructor");
            }
}
class Drawing extends Art{
```

```
    Drawing(){
        System.out.println("Drawing constructou");
    }
}
class Cartoon extends Drawing{
    public Cartoon(){
        System.out.println("Cartoon constructor");
    }
}
public class Test{
   public static void main(String args[]){
      new Cartoon();
}
}
```

2．抽象类实现接口。

```
interface A{
    public String author = "lili" ;
    public void printInfo() ;
    public String getInfo() ;
}
abstract class B implements A{
    public abstract void say() ;
}
class X extends B{
    public void say(){
        System.out.println("Hello World!!!") ;
    }
    public String getInfo(){
        return "HELLO" ;
    }
    public void printInfo(){
        System.out.println("作者: " +author) ;
    }
}
public class Test{
    public static void main(String args[]){
        X x = new X() ;
        x.say() ;
        x.printInfo() ;
    }
}
```